CLOUD MANAGEMENT BLOG

DNS Load Balancing and Using Multiple Load Balancers in the Cloud

October 23, 2012, Posted by Patrick McClory

Updated March 16, 2016

Load balancing in general is a complicated process, but there's some secret sauce in managing DNS along with multiple load balancers in the cloud. It requires that you draw from a few different sets of networking and “cloudy” concepts. In this second article in my best practices series (my first post covered how to use credentials within RightScale for storing sensitive or frequently used values), I'll explain how to set up load balancers to build a fault-tolerant, highly available web application in the cloud.

Here's what you’ll need:

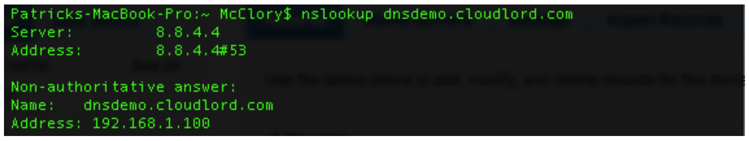
- Multiple A records for a host name in the DNS service of your choice
- Multiple load balancers to protect against failure

Before I explain how the two work together, let's check out how each of them works individually.

Multiple A Records for a Host Name

A records translate friendly DNS names to an IP address. For example, when you type rightscale.com in your browser, behind the scenes your computer is asking a DNS server to translate the name to an address.

I'm working from a Mac and the process is a little different for Windows-based machines, so check out more on nslookup for *nix and Windows. I'm also using one of Google's public DNS servers to perform my lookups. Check out the request below and note that when I query DNS, I'm getting a single address back for my test domain of dnsdemo.cloudlord.com:



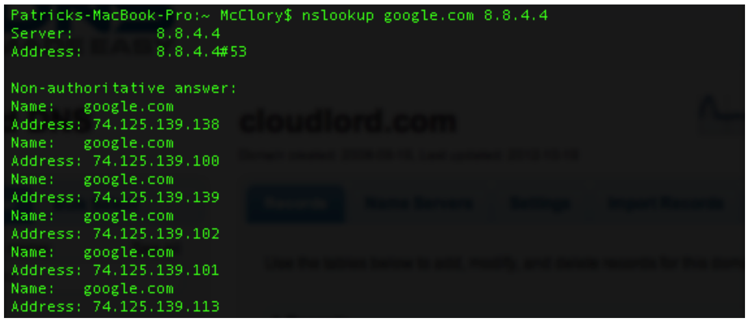
```
Patricks-MacBook-Pro:~ McClory$ nslookup dnsdemo.cloudlord.com
Server:         8.8.4.4
Address:        8.8.4.4#53

Non-authoritative answer:
Name:   dnsdemo.cloudlord.com
Address: 192.168.1.100
```

Figure 1 - DNS record with a single A record

My test domain has one A record associated and it resolves to the IP noted.

Let’s check out a more complicated example — Google.com:

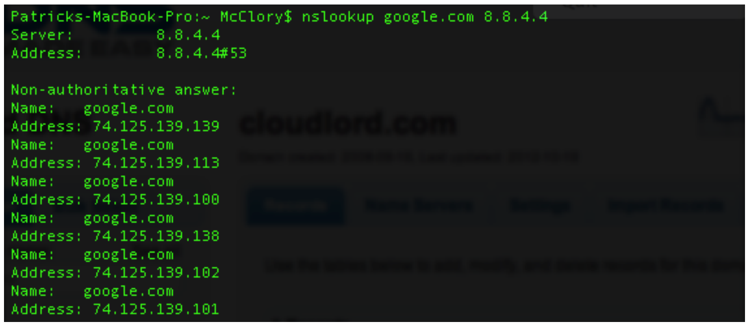


```
Patricks-MacBook-Pro:~ McClory$ nslookup google.com 8.8.4.4
Server:         8.8.4.4
Address:        8.8.4.4#53

Non-authoritative answer:
Name:   google.com
Address: 74.125.139.138
Name:   google.com
Address: 74.125.139.100
Name:   google.com
Address: 74.125.139.139
Name:   google.com
Address: 74.125.139.102
Name:   google.com
Address: 74.125.139.101
Name:   google.com
Address: 74.125.139.113
```

Figure 2 - DNS lookup of google.com showing six A records

Note that as shown in Figure 3 below, Google returned six addresses (at the time I queried it) because Google has six A Records registered to serve its main domain. When I ran the same nslookup query again, the IP addresses were returned in a different order. This is commonly referred to as “DNS load balancing.”

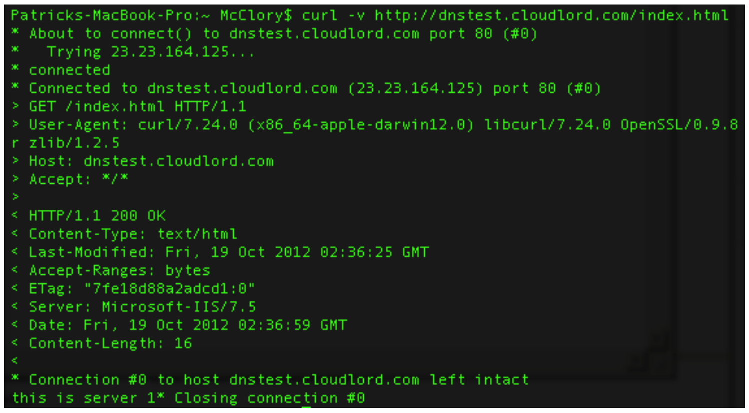


```
Patricks-MacBook-Pro:~ McClory$ nslookup google.com 8.8.4.4
Server:         8.8.4.4
Address:        8.8.4.4#53

Non-authoritative answer:
Name:   google.com
Address: 74.125.139.139
Name:   google.com
Address: 74.125.139.113
Name:   google.com
Address: 74.125.139.100
Name:   google.com
Address: 74.125.139.138
Name:   google.com
Address: 74.125.139.102
Name:   google.com
Address: 74.125.139.101
```

Figure 3 - DNS lookup of google.com showing A records in a different order than Figure 2

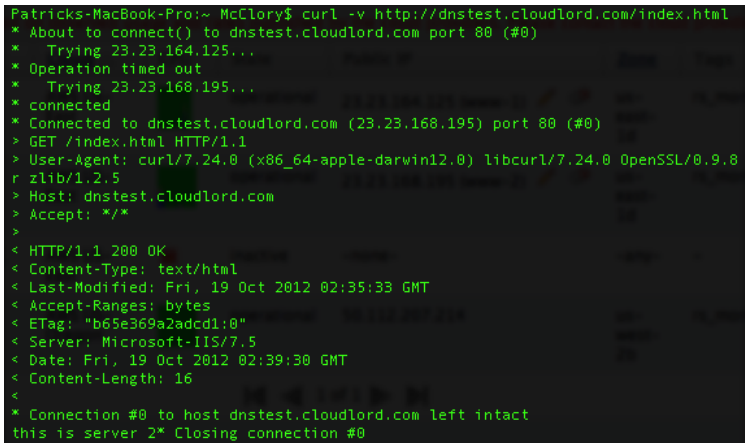
Figure 4 below shows DNS load balancing in action using dnstest.cloudlord.com and a test file that indicates which server is being served up. For this example, I set up my dnstest.cloudlord.com domain with two A records. Note that this first request has one attempt and the content reads that "this is server 1."



```
Patricks-MacBook-Pro:~ McClory$ curl -v http://dnstest.cloudlord.com/index.html
* About to connect() to dnstest.cloudlord.com port 80 (#0)
*   Trying 23.23.164.125...
* connected
* Connected to dnstest.cloudlord.com (23.23.164.125) port 80 (#0)
> GET /index.html HTTP/1.1
> User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0 OpenSSL/0.9.8
r zlib/1.2.5
> Host: dnstest.cloudlord.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html
< Last-Modified: Fri, 19 Oct 2012 02:36:25 GMT
< Accept-Ranges: bytes
< ETag: "7fe18d88a2adcd1:0"
< Server: Microsoft-IIS/7.5
< Date: Fri, 19 Oct 2012 02:36:59 GMT
< Content-Length: 16
<
* Connection #0 to host dnstest.cloudlord.com left intact
this is server 1* Closing connection #0
```

Figure 4 - Curl request to test domain with full availability of all servers

Next, I terminated the first server on the next request (to force a failed state), and the results are shown in Figure 5 below. Note that there's a timeout on the first IP, and then the second request goes through without any issue. You'll also notice that it's returning a response of "this is server 2."



```
Patricks-MacBook-Pro:~ McClory$ curl -v http://dnstest.cloudlord.com/index.html
* About to connect() to dnstest.cloudlord.com port 80 (#0)
*   Trying 23.23.164.125...
* Operation timed out
*   Trying 23.23.168.195...
* connected
* Connected to dnstest.cloudlord.com (23.23.168.195) port 80 (#0)
> GET /index.html HTTP/1.1
> User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0 OpenSSL/0.9.8
r zlib/1.2.5
> Host: dnstest.cloudlord.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html
< Last-Modified: Fri, 19 Oct 2012 02:35:33 GMT
< Accept-Ranges: bytes
< ETag: "b65e369a2adcd1:0"
< Server: Microsoft-IIS/7.5
< Date: Fri, 19 Oct 2012 02:39:30 GMT
< Content-Length: 16
<
* Connection #0 to host dnstest.cloudlord.com left intact
this is server 2* Closing connection #0
```

Figure 5 - Curl request to test domain with primary A record in failed state (note timeout and new IP)

The order in which IP addresses are returned varies by the DNS server and provider used but often follows a round-robin or geographically specific algorithm.

The idea here is that different clients will get different ordered lists of IP addresses corresponding to your domain name. This has the effect of distributing requests across the group of IPs in a specific manner. If an IP address is does not respond in an appropriate amount of time, the client will time out on that request and move on to the next IP address until the list is exhausted or it finds a connection that's valid. Although it's not an exhaustive list, most modern browsers, along with curl as shown above in Figure X, follow this retry process.

There are a few things to remember though:

- DNS failover doesn't provide any additional features such as “sticky sessions” for your application.
- Upstream DNS caching is unpredictable — client DNS providers may or may not respect your TTL settings.
- This isn't a replacement for TCP load balancing because it's not terribly precise based on the upstream DNS caching process noted above.

Multiple Load Balancers for Redundancy and Scalability

With multiple IP addresses routing to your deployment, each of these addresses can terminate at a load balancer that serves your back-end application (see Figure 6 below). Doing this, you'll be able to present multiple endpoints to the public to serve your application (I'll get back to why this is important in a minute).

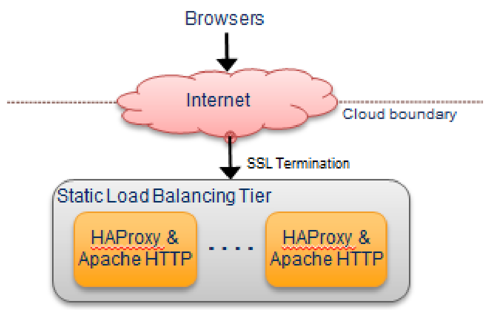


Figure 6 - Overview of incoming connections from public Internet to TCP load balancers

In Figure 7 below, I go a step further and illustrate how connectivity to the application layer can be set up from multiple TCP load balancers. This allows you to have multiple incoming connections each serving up the same content, providing a redundant load balancing layer as well as a redundant application layer.

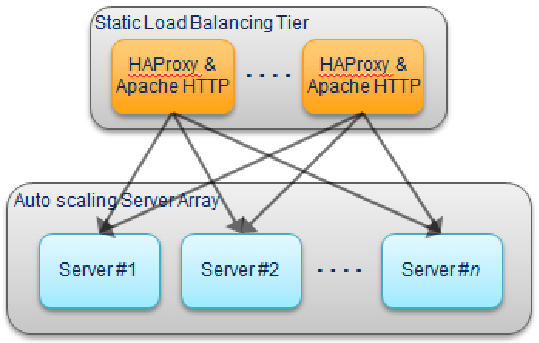


Figure 7 - Connection diagram for multiple load balancers connecting redundantly to the same application server tier

DNS Load Balancing: Bringing It All Together

The end result is that by using DNS load balancing, you can achieve a fairly rough balance of traffic between multiple TCP load balancers, which can manage applying load to your application servers at a more granular level:

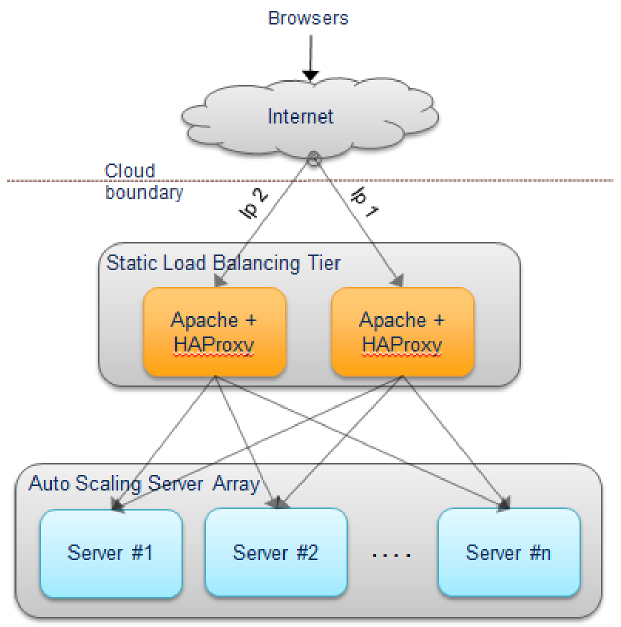


Figure 8 - Full incoming connection diagram showing multiple load balancers with their own IP address

This is a great way to protect against failure and increase overall throughput, giving you the ability to scale for high availability and high performance.

Setting up DNS load balancing can be a bit of a hassle, but the Load Balancer with HAProxy ServerTemplate, along with scripts for application servers to attach to load balancers, simplifies the process. The RightScale ServerTemplate™ and scripts use a tag-based, managed solution that will keep your HAProxy config files synchronized and that will automate the deployment, registration, and detach process for all servers involved. To use the ServerTemplate for setting up DNS load balancing, sign up for a free RightScale trial.

Read more articles on Load Balancing:

Benchmarking Load Balancers in the Cloud

Popular Posts

Cloud Computing Trends: 2017 State of the Cloud Survey

Cloud Cost Allocation with RightScale Optima Billing Centers

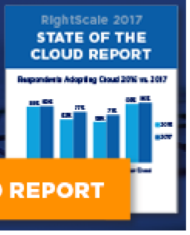
AWS vs Azure vs Google Cloud Pricing: Compute Instances

AWS Costs: How Much Are You Wasting?

How to Compare AWS vs Azure vs Google vs SoftLayer

How Telcos and Other MSPs Are Succeeding in Cloud

Private and Hybrid Clouds: 9 Use Cases and Implementation Advice



Tags

AWS Cloud Analytics Cloud Computing Cloud Computing Best Practices cloud cost management **Cloud Costs**

Cloud Management EC2 Rackspace **RightScale** RightScale Product Updates ServerTemplate

Top Authors

Kim Weins

Thorsten von Eicken

Brian Adler

View All Authors

3 Comments RightScale Blog Patrick McClory

Recommend Share Sort by Newest

Join the discussion…

rspatrick • 5 years ago

Hi Jon,

I certainly agree with you on the sub-optimal client experience and I don't think that the goal is to stay in the 'failed' state for a long period of time, rather to keep the site/service up and running while recovery is underway. I also worked this out using an existing DNS provider-I generally don't set up DNS load balancers for cloud deployments, but it's definitely something worth checking out. My main concern about a single 'A' record is that once that server fails, there needs to be a plan to manage the failure of the DNS load balancer, so the other goal of two ingress IP addresses (in this case) was to mitigate the risk of either one of them going down. Setting up a redundant DNS infrastructure across disparate zones/cloud regions or providers could also be an option.

Best,

Patrick

Reply • Share ›

Jon Braunhut → rspatrick • 5 years ago

Hi Patrick,

I'm with you on both points. I don't think many folks are (or should be) engineering for an extended "failed" state. And if someone is going to pursue the single "A" record approach, it's that much more important to have redundant DNS infrastructure. But most people with serious deployments will already have planned for DNS redundancy, so there's little marginal cost to going down this road.

Cheers,

Jonathan

Reply • Share ›

Jon Braunhut • 5 years ago

Hey Patrick,

Nice post! You've raised some great issues, and the multi-tier scheme you've illustrated in figure 8 is a solid design template. Just two quick observations:

1) Regarding TCP load balancer failure, while it's true that most browsers do have reliable "time out and move on" behavior, that can still result in a suboptimal client experience. An alternative to consider might be a DNS load balancer that does active healthchecking of the TCP lb's in the first place, to avoid clients having to time out at all.

2) With a DNS load balancer in place that does that healthchecking, it's possible to deliver single "A" records to clients, instead of multiples. And it's then possible to maintain a "sticky" association between the client and its TCP load balancer instance. For delivering apps that are more stateful, this is an invaluable tweak.

Cheers,

Jon Braunhut

Chief Scientist

KEMP Technologies

• Reply • Share ›

ALSO ON **RIGHTSCALE BLOG**

How to Compare AWS vs Azure vs Google vs SoftLayer

1 comment • 2 years ago

Shivam Parikh — What is missing that Sofltayer includes bandwidth free packages such as 500GB, 1TB, 10TB …

AWS vs Azure vs Google Cloud Pricing: Compute Instances

6 comments • a year ago

Tom Pauwaert — Really great article. Thanks very much for the work! It makes it a lot easier to compare and to figure …

Cloud Storage: AWS vs Azure vs Google vs SoftLayer

1 comment • 2 years ago

Khurram — Hello, could you elaborate on the SLA's for block storage?

Migrating to Docker: Why We're Going All in at RightScale

2 comments • 2 years ago

rstimmiller — Hi Giri, Yes, that is what our best estimates were telling us. When the dust settles and we see where we …

Subscribe **Add Disqus to your siteAdd DisqusAdd** **Privacy**

CLOUD MANAGEMENT BEST PRACTICES

CLOUD INDUSTRY INSIGHTS

ENTERPRISE CLOUD STRATEGIES

CLOUD COST ANALYSIS

RIGHTSCALE NEWS